This information is intended for those who have a working knowledge of Public Key Infrastructure. The paper assumes minimal knowledge of digital timestamping and as such includes a Reference section at the end.

Executive Summary

Trust is an inherent part of a PKI solution. That trust, however, must be *verifiable*. Consequently, a PKI uses digital certificates to verifiably associate a person with a specific public key. In turn, timestamping plays a complementary and critical role in a PKI by verifying digital certificates and digital signatures.

- Checking the Validity of Certs: to prove that a digital certificate (cert) has not expired, or been revoked, it must be compared to a timebase. If the time values are not secure timestamps, the user is vulnerable to being spoofed by a maladjusted system clock.
- Ensuring Non-repudiation of Digital Signatures: when a digitally signed record must live beyond the validity period of its cert, a secure timestamp can prove that the record was signed *during the validity period of the cert.*

It is clear that any company that relies on an internal system clock for a timestamping solution may be challenged to a *motive and means* argument in a legal dispute. A company that issues its own timestamps, is no different from a company that audits its own financial information. It is highly suspect and simply isn't done. However, it is not as obvious that a trusted third party timestamping system is open to collusion -- either through bribery, or coercion -- and thereby is also vulnerable to a motive and means legal attack.

- Hash & Sign Timestamping: the third party has only to capture digital time signals for future use -- in essence creating false time. Then the third party uses the false time with the hash value of a document supplied by the client and signs it. Of course it can be cryptographically validated, but it is a document with a false timestamp.
- Secure Hardware Timestamping
    - A physically isolated time source may prevent a network attack from changing the timebase, but the time source itself must be periodically conditioned against an external time source to maintain accuracy, and is hence still vulnerable to collusive attacks by insiders.
    - Similarly, a physically-secured crypto module may prevent an external attack from directly affecting the cryptographic processing of the timestamp. However, the module must have means for setting its private key by the owner, which again leaves the system vulnerable to collusive attacks by insiders.

In fact, using *keyed* cryptography for timestamping is extremely flawed. The cert used by the third party to sign the timestamp will eventually end up on a Certificate Revocation List -- either by expiration, or revocation. When that happens, you will not be able to prove that your document was signed during the validity period of your own

cert. This inherent problem defeats the very purpose of timestamping a digitally signed document.

The solution to collusive attacks is a *keyless* timestamping system that is not open to insiders and that makes the timestamping publicly witnessed so that nothing can be altered retrospectively without its being detected.

- The requestor hashes the document and sends the hash value to a third party, who supplies digital time -- as in other third party models.
- The third party supplies time and date independently.
- The document's hash value is made part of a hash chain by including it with other hash values submitted for timestamping by various clients. Those hash values are passed through a one-way hashing algorithm, yielding a summary hash value dependent upon all the hash values.
- A timestamping record is returned to the requestor. It includes the place in the hash chain occupied by the requestor's document at the exact moment of timestamping and the other hash values, which were submitted for timestamping by various clients at that same time. It also includes the time and date of the timestamping.
- The summary values of the hash chain are made widely public. The hash chain cannot be retrospectively altered, or faked, without detection, so the third party does not have to be trusted.
- Validating the document's content and time/date is done by creating a test summary hash value from the information included in the timestamping record. The test value is then compared to the real summary hash value, which is found in the public hash chain. If the test value and the real value are the same, it is indisputable that this exact digital record was timestamped at exactly this moment.

In summary, secure timestamping can improve the trustworthiness of a PKI by applying it to: certificates, certificate revocation information, audit logs, audit trails composed of checking the CRL and the signed document, and providing non-repudiation services for signed documents.

Timestamping in a PKI

Trust is an inherent part of a PKI. However, trust is not blind faith. The goal in a PKI is *verifiable trust*. Consequently, a PKI uses digital certificates to verifiably associate a person with a specific public key. In turn, timestamping plays a complementary and

critical role in a PKI by substantiating the validity of digital certificates and digital signatures.

In Checking the Validity of Certs...

In the event of a dispute, it is critical to prove that you checked the validity of the cert upon which you relied. Otherwise, your reliance upon it would be *unreasonable* in a court of law.[1] (Of course, it's just common sense to check the validity anyway.)

To prove a cert has not expired, the validity period values must be compared to some timebase. If those time values are not secure timestamps, the user checking the validity of a cert is vulnerable to being spoofed by a miscalibrated system clock, and hence is in danger of accepting a cert as valid that has in fact expired.

To prove a cert has not been revoked, the current time must be compared with the timestamp on the Certificate Revocation information (either in a CRL, or real-time registry) obtained from the CA. Again, if the various times being compared are not secure timestamps, the user checking the validity of the cert is vulnerable to maladjusted clocks.

In Building Non-repudiation Services for Digital Signatures...

A digital signature is only valid while within the validity period indicated in the cert used to compute the signature. However, there are many instances where a signature must live for a potentially long period of time, for example, when certs are used to sign contracts or building plans or securities trades or.... In these cases there must be an additional mechanism in place for extending the lifetime of a signature beyond the validity period of the cert.[2]

The way to do this is secure timestamping. If the signer of a document (any digital record) also timestamps enough additional information, the signature can be validated successfully long after the cert has expired. This is the essence of what we mean by a non-repudiation service.

To extend the life of a signature, a number of data items must be timestamped. The document itself must be timestamped, along with the signature in question; this is to protect against future attacks on the cryptographic primitives used to form the signature. The cert used to compute the signature, and all certs used to sign those certs must also be timestamped, to prove they were used during their respective validity periods. The CRL information for all certs involved must also be timestamped, to prove that the certs

[1] American Bar Association, *Digital Signature Guidelines,* Guideline 5.4 (Reasonableness of Reliance). . www.abanet.org/scitech/ec/isc/dsgfree.html

[2] *Digital Signature Guidelines,* Section 1.20.

were not revoked at the time the signature was formed. If all that information is timestamped with a secure timestamping mechanism, it is possible to prove the time of the signature and that it was generated by a valid cert – long after the cert itself has expired.

Timestamping 101

A Secure Timestamping mechanism is one where it is impossible for either the original document to be altered, or a timestamp falsified, without detection.

Insecure Timestamping Methods

Internal System Clocks as the Time Source

It is obvious you have the means to fake the time when it is under your control. This ability enables you to substitute false, signed documents for originals, or backdate signed documents, as required.

Trusted 3rd Party Timestamping

Timestamping systems based on trusted third parties are inherently vulnerable to a number of different security attacks. Third party systems are vulnerable to collusion, e.g. providing unfair advantages to competing clients in exchange for monetary gain. Third parties are also vulnerable to spoofing attacks, e.g. an adversary presenting a service indistinguishable from the real service by a trusting client.

Hash & Sign Timestamping

The document is first processed by a one-way hashing algorithm to protect its confidentiality. The resultant hash value[3] is sent to a third party service for timestamping. The timestamping service supplies a digital time record on receipt of the hash value, which it concatenates with the hash value. The resultant bit string of hash value and time is then signed by the timestamping service. The result is sent back to the owner of the record. It can be validated in the future in the same manner as validating any digitally signed record.

[3] The hash value of a specific digital record is unique to the bits of that record. The probability of being able to create a different digital record with the same hash value as the original record is 1 in 10^{87}. You would have a better chance at guessing the location of an atom in the universe. (There are 10^{80} atoms in the known universe.)

However, this method is vulnerable to collusive attacks. The third party can be persuaded either through bribery, or coercion, to cooperate in backdating a timestamp.

No matter what time source is used, the third party has only to capture digital time signals for future use -- in essence creating false time. Then the third party uses the false time with the hash value of a false document supplied by the client and signs it. Of course it can be cryptographically validated, but it is a false document with a false timestamp.

Secure Hardware Time-stamping

Vendors of third party services have tried to ameliorate their problems by fortifying their services with specialized hardware. Some timestamping services maintain their own atomic time clocks, which are periodically synchronized with the official NIST (National Institute of Standards) time source through some out-of-band means. In theory this prevents attackers from adjusting the time source in order to forward date or back date digital information.

Other services use specially sealed hardware cryptographic modules, which are protected from tampering by encasing the crypto engine in a tamperproof, sealed barrier. In theory this prevents an attacker from being able to affect the operation of the cryptographic module.

Some timestamping service implementations even go so far as encasing both the local atomic time source and the sealed cryptographic module in another sealed hardware module, to prevent an attacker from interfering with the interactions between those modules.

These hardware-related barriers to tampering provide an additional degree of security to such a system, but in reality they do nothing to address the fundamental problems with trusted third party systems.

- A physically isolated time source may prevent a network attack from changing the timebase, but the time source itself must be periodically conditioned against an external time source, and is hence still vulnerable to collusive attacks by insiders.

- Similarly, a physically-secured crypto module may prevent an external attack from directly affecting the cryptographic processing of the timestamp, however, the module must have means for setting its private key by the owner, which again leaves the system vulnerable to collusive attacks by insiders.

- Finally, cryptographic keys used in physically isolated modules have finite lifetimes just as other cryptographic keys do, and hence must be changed periodically. This makes hardware-assisted timestamping services as unappealing for use in real world applications as other timestamping services based on keyed cryptography.

Timestamping with Keyed *Cryptography -- Inherently Flawed*

Using keyed cryptography for timestamping is flawed, because the cert used by the third party to sign the timestamp will eventually end up on a CRL -- either by expiration, or revocation. When that happens, you will not be able to prove that your document was signed during the validity period of your own cert. This inherent problem defeats the very purpose of timestamping a digitally signed document.

A possible solution to the problem of the timestamping service's cert expiring is, prior to its expiration, to re-timestamp the signed records using a cert that expires at a later date. This approach sounds good in cryptographic theory, but does not work very well in the real world.

- For example, a mutual fund company has millions of customers and hundreds of millions of transactions with those customers during a year. Considering that the company is required to keep its records for seven years, that's billions of records to keep track of to determine which ones need re-timestamping at what points in time.

  So, how about simply hashing each archive tape, get digital time independently, and sign the hash/time bit string? That will certainly work cryptographically, but not in the real world of legal discovery.

  In a legal dispute between the company and one of its customers, the company's counsel will not enter into evidence all the records on the archive tape for two reasons:

  1. There may be evidence included on that tape that will reflect negatively on their case. That's why documents are carefully reviewed during a discovery request to determine if they are responsive, non-responsive, or privileged (between attorney and client). Only responsive documents are turned over to plaintiff's counsel.

  2. There will be records of other customers on that tape who are not involved in the dispute, but whose records are private. (Those records would be part of the hash value of the whole archive tape, but could not be turned

over. Therefore, the timestamp for the archive tape could not be validated when challenged.)

This example could just as well be a pharmaceutical company vitally concerned with proving its intellectual property rights, which depend on millions of digital documents and records created by their scientists in the course of inventing a new compound. Or, a government agency that is accused of tampering with its electronic bidding process in order to award contracts to "favored" suppliers. The problem is the same in any case.

Secure Timestamping Method -- the Widely Witnessed *System*

The problem of collusive attacks can be solved through a system that is not dependent on *keyed* cryptography and that makes the timestamping publicly witnessed so that nothing can be altered retrospectively without its being detected.

In a widely witnessed system, the requestor hashes the document and sends the hash value to a third party, who supplies digital time -- as in other third party models. The rest of the process is completely different.

- The third party supplies time and date independently, but does not digitally sign that bit string, because this is a keyless technology.

- Rather, the document's hash value is made part of a hash chain by including it with other hash values submitted for timestamping by various clients. Those hash values are passed through a one-way hashing algorithm. The result is a summary hash value dependent upon all the hash values.[4]

- A timestamping record is returned to the requestor. That timestamping record includes the place in the hash chain occupied by the requestor's document at the exact moment of timestamping and the other hash values, which were submitted for timestamping by various clients. It also includes the time and date of the timestamping.

- The hash chain is made widely public. The hash chain cannot be retrospectively altered, or faked, without detection. The third party does not have to be trusted, because the one-way hashing technology precludes tampering.

- Validating the document's content and time/date is done by rehashing the document and using that hash value along with the other hash values included in

[4] The probability of being able to retrospectively substitute a false document's hash value and then produce the same summary hash value while still retaining the other chain hash values is far to the right of infinitesimal.

the timestamping record to create a test summary hash value. The test summary hash value is then compared to the real summary hash value. The real summary hash value is found in the publicly published hash chain. If the test value and the real value are the same, it is indisputable that this exact digital record was timestamped at exactly this moment.

(Surety.com's award winning timestamping process can be viewed in detail at http://www.surety.com/i-technical/sld001.htm. The URLs for the scientific papers are included in the *References* section.)

Secure Timestamping in a PKI

The areas in which secure timestamping should be applied are:

- CA Operations
    - Certificates: when they are issued, renewed, revoked, resigned, or when an attribute changes.
    - Certificate revocation information.
    - Certificate status checking mechanisms: audit logs, CRLs, OCSP repositories, etc.
- Specific Applications (any which include digital signing)
    - Audit Trail: when checking a digital signature, create an audit trail of checking both the CRL and the document and then timestamp that audit trail. An irrefutable audit trail is critical in the event of a dispute to prove that it was reasonable to rely upon the signature.
    - Extend the Life of the Digitally Signed Document: timestamp the document after signing so that it can be proven that the document was signed during its cert's validity period after that cert has expired, or been revoked. (Please see http://www.rsa.com/rsalabs/faq/html/7-11.html.) This application of secure timestamping is very important with records that must be relied upon for the extended periods required by regulatory agencies such as the SEC, FDA, EPA, USPTO, FAA, etc.

Surety Integration Tools

The following tools can be used either by you, or by a third party's professional services group on your behalf, to facilitate the integration of Surety's timestamping service into your PKI.

- Software developer's kit (SDK): allows a programmer to embed Surety technology in an existing application for seamless operations. This SDK contains C++ libraries, documentation and a sample program.

- Integration kit for Microsoft developers: provides full access to the Digital Notary Service from applications written using Visual Basic, Visual J++, and other COM/ActiveX-enabled development tools.

- Integration kit for Lotus developers: an LSX that provides: provides full access to the Digital Notary Service from applications built for Lotus Notes and Domino.

(Please note that it is not necessary to start up your PKI with Surety timestamping: it can be turned on via administrative configuration when you are ready.)

Interoperability

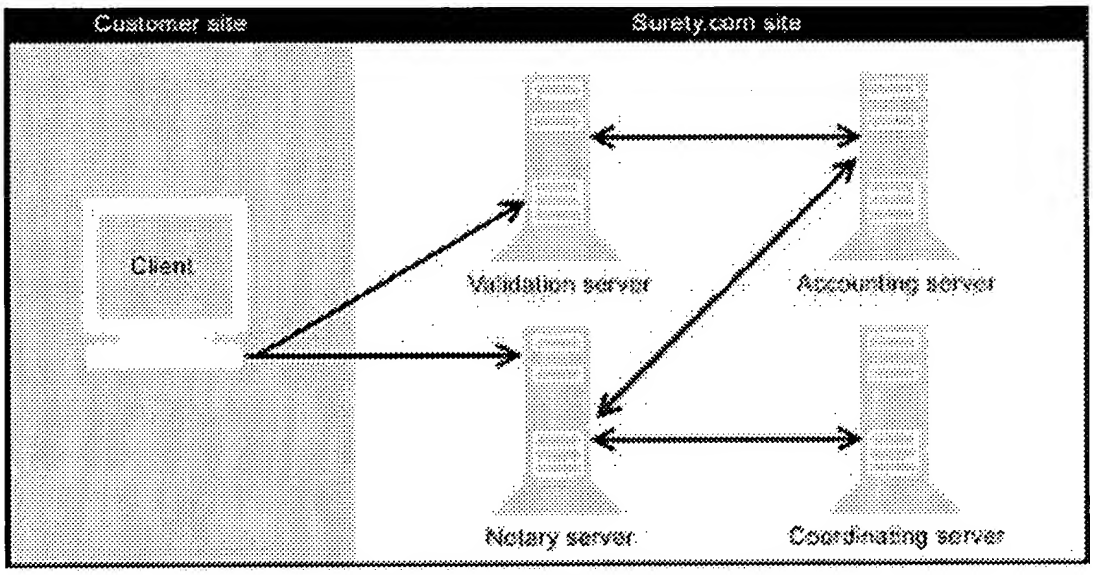
Surety's secure timestamping uses standards to ensure interoperability.

- Communication: TCP/IP

- Time: UTC timebase (Since radio signals can cross multiple time zones and the international date line, some worldwide standard for time and date is needed. This standard is **coordinated universal time**, abbreviated **UTC**. This was formerly known as **Greenwich Mean Time**, or GMT). Time comes from a GPS-conditioned clock, which resolves time to within 100 nanoseconds.

Digital Notary Service Architecture

Surety's Digital Notary Service is modularly architected with no single point of failure. While the process itself is patented, it uses Internet standards for communication and security.



Digital Notary Service Architecture (typical)

Client

The client software hashes the document(s) to be timestamped, issues a request for timestamping services and connects to a Notary server. The client is actually an API, which is integrated in a specific application.

Notary Server

A Notary Server is the software component contacted by a client to request timestamping services. The Notary Server performs the following steps:

1. The Notary Server contacts the Accounting Server to ensure the username and password specified in the timestamping request reference an active account.

2. The file hash value submitted in the timestamping request is added to a group of other file hash values submitted by other clients.

3. When the group reaches a set size or a set time quanta expires, the Notary Server calculates a binary tree of the file hash values in the group.

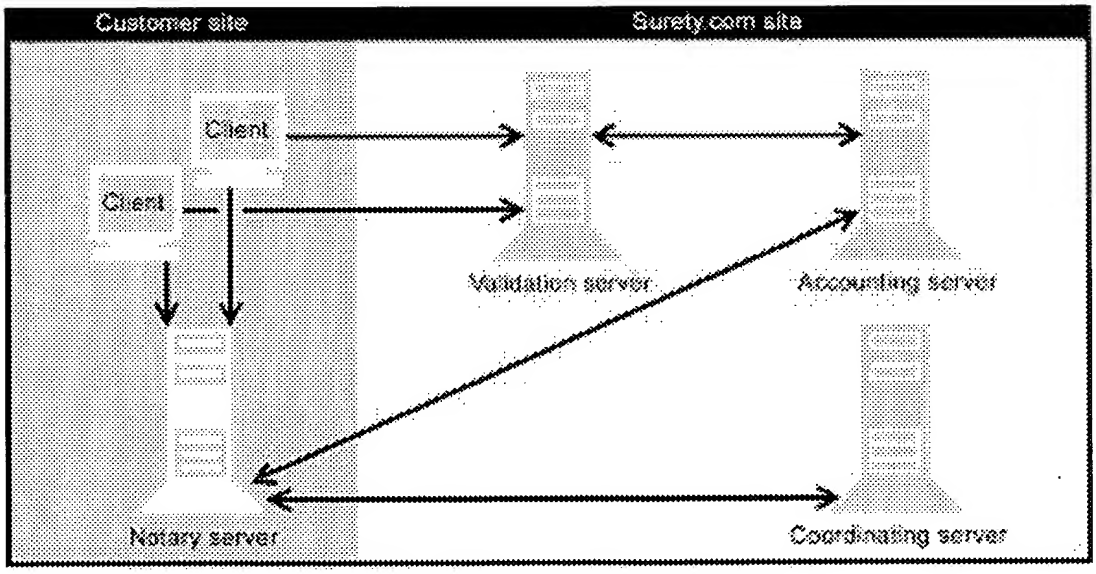
4. The super hash value at the top of the calculated tree is submitted to the Coordinating Server for linking and timestamping.

5. The linked super hash values and timestamp returned from the Coordinating Server are combined with the intermediate hash values, and returned to the requesting client.

6. The requesting client is billed for the timestamping transaction.

Notary Servers act as concentrating points for timestamping requests to the Digital Notary Service. There are a number of publicly available Notary Server nodes maintained by Surety.com and dispersed throughout the Internet for use by any Internet-connected user. By concentrating timestamping requests at Notary Servers, the load on the Coordinating Server is significantly reduced.

Additionally, large organizations anticipating high volumes of timestamping requests, those with special security needs, or those who desire greater control over the location, behavior, and output of a Notary Server have the option to purchase a Notary Server for installation locally at their site.



Digital Notary Service Architecture (high-volume)

By maintaining multiple identical Notary Server nodes throughout the Internet, Surety.com provides clients with timestamping capability even in the face of network congestion or outages. Surety.com maintains a web server at the central office containing the current list of publicly available Notary Server nodes.

Coordinating Server

A Coordinating Server is a software component contacted by Notary Servers to request linking and timestamping services. The Coordinating Server is the means whereby timestamping requests from individual users are linked to timestamping requests from all other users of the service. The Coordinating Server handles requests from Notary Servers only, and is not directly visible to the user. Linking together all timestamping requests by all users serves to spread the trust in the system among all users of the system.

The Coordinating Server performs the following steps.

1. The Coordinating Server links the super hash value submitted by a Notary Server to the latest linked super hash value.

2. The newly linked super hash value is written to an online stable repository along with the current high-precision time.

3. The linked super hash and time-stamp values are returned to the requesting Notary Server.

4. The latest linked super hash value is replaced by the recently returned super hash value.

5. Periodically, the online stable repository is moved to offline archival storage in bulk.

Surety.com maintains a small set of distinct Coordinating Servers installed on hardware in dispersed locations, and configured to gracefully failover in the event of network congestion or outages.

Validation Server

A Validation Server is the software component contacted by a client to request validation services. The Validation Server performs the following steps:

1. The Validation Server contacts the Accounting Server to ensure either the user requesting the validation or the user who created the Notary Record has an active account.

2. The contents of the Notary Record submitted in the validation requests are used to calculate the binary hash tree and linked super hash value for the given file.

3. Requesting client is notified if the calculated super hash value and time-stamp matches the record of super hash values and time-stamps stored in the repository.

Validation Servers act as concentrating points for validation requests to the Digital Notary Service. As with Notary Servers, there are a number of publicly available Validation Server nodes maintained by Surety.com and dispersed throughout the Internet for use by any Internet-connected user. As demand for the Digital Notary Service grows, Surety.com will respond to increased validation load by deploying additional Validation Server nodes. Surety.com maintains a web server at the central office containing the current list of publicly available Validation Server nodes.

Accounting Server

An Accounting Server is the software component contacted by Notary Servers and Validation Servers to request user account and billing information.

The Accounting Server performs the following steps:

1. If the username exists in the accounting database, account balance information is returned to requesting Notary Server or Validation Server nodes.

2. Balance updates from Notary Server and Validation Server nodes are coordinated and incorporated into the account database as needed.

The Accounting Server provides infrastructure support to other components in the Digital Notary Service, and as such is not directly visible to clients. Surety.com maintains the Accounting Server, installed at the central office. Both Notary Servers and Validation Servers are structured to continue operating for a given period of time even if the Accounting Server is out of touch.

Additionally, Notary Servers and Validation Servers are structured to take advantage of user locality to cache accounting data; e.g. users are likely to use the same Notary or Validation server for successive timestamping or validation requests. These two techniques are employed to both lower the load on the Accounting Server, and to greatly reduce the chances of the Accounting Server becoming a single point of failure.

References

Subject	Source
Hash Function	http://www.rsa.com/rsalabs/faq/html/2-1-6.html
Digital Timestamping Overview	http://www.rsa.com/rsalabs/faq/html/7-11.html
How to Digitally Timestamp a Document	http://www.surety.com/papers/1sttime-stampingpaper.pdf
Improving the Efficiency and Reliability of Digital Timestamping	http://www.surety.com/papers/BHSpaper.pdf
Legal Strengthening of PKI with Digital Timestamping	http://www.pkilaw.com/surety_time_long_3.htm